The Package-Based Development Process in the Flight Dynamics Division

Amalia Parra, Computer Sciences Corporation
Carolyn Seaman, University of Maryland
Victor Basili, University of Maryland
Stephen Kraft, NASA/Goddard Space Flight Center
Steven Condon, Computer Sciences Corporation
Steven Burke, Computer Sciences Corporation
Daniil Yakimovich, University of Maryland

Abstract

The SEL has been operating for more than two decades in the FDD and has adapted to the constant movement of the software development environment. The SEL's Improvement Paradigm shows that process improvement is an iterative process. *Understanding*, *Assessing* and *Packaging* are the three steps that are followed in this cyclical paradigm. As the improvement process cycles back to the first step, after having *packaged* some experience, the level of *understanding* will be greater. In the past, products resulting from the *packaging* step have been large process documents, guidebooks, and training programs. As the technical world moves toward more modularized software, we have made a move toward more modularized software development process documentation, as such the products of the *packaging* step are becoming smaller and more frequent. In this manner, the QIP takes on a more spiral approach rather than a waterfall.

This paper describes the state of the FDD in the area of software development processes, as revealed through the *understanding* and *assessing* activities conducted by the COTS study team. The insights presented include: (1) a characterization of a typical FDD COTS intensive software development life-cycle process, (2) lessons learned through the COTS study interviews, and (3) a description of changes in the SEL due to the changing and accelerating nature of software development in the FDD.

1 Background

The Flight Dynamics Division at NASA/Goddard Space Flight Center has had a history of effective reuse of software to levels as high as 90%. The increase has been affected by the use of Ada and object-oriented technologies. This experience led to the creation and use of an architectured component library for a certain class of systems so that these systems could be "configured" rather than developed [Condon et al., 1996]. It has also motivated the outsourcing of software development for more "standard" systems, which in turn has led to a move from internal reuse to the use of external software packages. The introduction of package-based software development—rapid configuration of software systems based on Commercial-Off-The-Shelf (COTS) packages, Government-Off-The-Shelf (GOTS) packages, and some custom-built reusable packages—has motivated the

Software Engineering Laboratory (SEL) to provide guidance in this new era by updating the *SEL Recommended Approach to Software Development* [SEL, 1992]. Before updating this important SEL guidebook, however, it was first necessary to understand and improve this new package-based process within the flight dynamics domain.

The traditional SEL approach to software improvement involves three steps. These are described in more detail in Section 2, but briefly they are as follows: (1) *understand* the current situation in the local environment (for us, the FDD) and develop appropriate goals for improving specific items; (2) *assess* how to achieve these goals by defining process changes, testing them on one or more projects, and analyzing the results of this experiment; (3) package the lessons learned from step 2 and integrate these into the local software development process. In any given SEL experiment, this 3-step improvement process is generally a cyclic one, involving several iterations. In addition the steps can overlap somewhat.

The first phase of the SEL COTS study was conducted during the last few months of 1995. Because the FDD had limited experience developing COTS-based systems at that time, the SEL looked at experiences of outside organizations in order to understand the challenges associated with this type of development and to gather best practices used on COTS-based projects. Using a solid understanding of the FDD project domain, history and environment, the SEL synthesized this information into a strawman process to be used to produce COTS-based systems in the FDD. This initial strawman process was then reviewed for feasibility by key FDD software engineers (both civil servant and contractor) who have had some experience with COTS. The resulting strawman process, presented in the *Packaged-Based System Development Process* [Waligora, 1996], is available on the SEL's Web page, http://fdd.gsfc.nasa.gov/selres.html.

As more FDD projects began using COTS to construct their software systems, the next phase of the SEL COTS study began. The goals of this phase, which is the subject of the rest of this paper, include gathering a current understanding of COTS-based project, suggesting areas of improvement for further study, and providing guidance to current and future COTS-based projects.

In section 2, we present some of the terminology used in the rest of the paper. Section 3 describes the approach we used in the study, and section 4 describes the COTS-based software development process that emerged from the data we collected, as well as some insights into that process. Section 5 describes some of the steps that the SEL has taken to keep up with the pace of change, in particular in packaging the results of the COTS study in a timely and relevant manner. Section 6 describes some of the future plans for this line of investigation.

2 Terminology

The words “Commercial-Off-The-Shelf” are very generic; they can be used to in reference to many different types and levels of software, e.g. software that fills a specific functionality or a tool used to generate code. In this paper the term *COTS* implies a COTS product that has specific functionality as part of a system — not merely a tool, but a piece of ‘pre-built’ software that is *integrated* into the system and must be *delivered* with the system to provide operational functionality or sustain maintenance efforts.

The term *COTS project* refers to a project that integrates COTS packages and other software to develop a system. This is not to be confused with the development of COTS packages that occurs at the ‘vendor’ corporation.

Additionally, the term GOTS is equivalent to COTS in this study because the process followed is for the most part identical to developing a system with COTS.

3 Experimental Approach

The Improvement Paradigm, shown in Figure 1, is a SEL tool for process improvement and is commonly used to plan SEL studies. The SEL COTS study team used this concept to guide its work. The paradigm is a three step, iterative process. The basic step is understanding, which identifies the current status of some aspect of software development in the SEL. The next step is assessing, which determines potential improvements. The main activities of the COTS study team are primarily focused on this assessing step. Packaging is the top step, in which improvements are documented and integrated into the environment to form the basis for the next level of understanding.

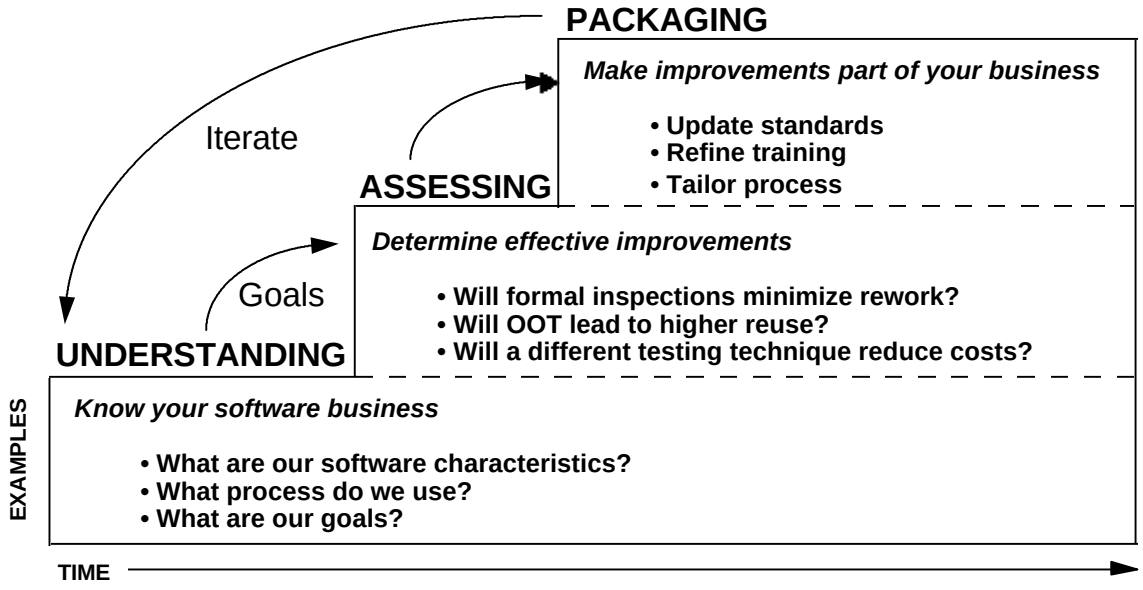

Figure 1. The Improvement Paradigm

The initial understanding step for the COTS study was a series of interviews with representatives from 12 COTS projects. Based on the interview data, we then described the COTS-based development in the FDD, presented in section 4.1. Also based on interview data, we redesigned the way development effort data is collected in the FDD. These two activities made up our assessing step. These results were packaged, in the packaging step, through updated data collection forms (described in section 5.2) and study briefs (described in section 5.3). The next level of understanding, then, is provided by the baseline process description and data from the new effort forms.

4 COTS-based Software Development Process

As a first step in understanding where COTS-based development in the FDD stood, the study team analyzed the current data collection. Historically the SEL collects effort data. For typical pre-COTS era projects the SEL has a baseline of effort divided into four simple categories of activities. The SEL anticipated need for data specific to COTS projects, made an attempt to gather data on this effort, but the level of detail was too general to allow understanding of the COTS-related effort. One indication that the SEL was not capturing useful data is the large amount of effort that fell into the "other" category.

Clearly, the quantitative information available was not sufficient for us to identify and understand the new issues that were arising in relation to the use of COTS packages in FDD projects. In order to gather more and richer information on this topic, the study team designed and conducted structured interviews, using three levels of interview guides at increasing levels of detail, with representatives from 12 projects. Topics covered included the process steps carried out, what problems were encountered with the use of COTS in development, and how the incorporation of COTS has changed the software development process.

4.1 Process Description

Our interviews uncovered the new process flow, shown in Figure 2. The study team discovered more complexity in the current practice than expected in theory. For example, we had expected vendor interaction to be simple, and to end with the purchase of a product. In reality, the interaction continues throughout the life cycle and the flow of information is not merely one way. Surprisingly, we found a strong dependence on *bi-directional* information flow. Also shown is a more constant involvement with separate organizations, such as other projects that also use COTS, independent evaluation teams, and other customers of the vendor. Portions of the COTS-based systems include traditional developed software. So an issue to consider is how to fit together our traditional process, as documented in the SEL Recommend Approach to Software Development, and our new way of doing business by integrating COTS packages to build a system.

The software development teams interviewed included both FDD and CSC personnel. Although not every team followed all of the steps outlined below, a composite process flow emerged from the interview data. Note: None of the project teams interviewed had

begun sustaining engineering. This step will be evaluated in future studies. The steps in the overall process, as shown in Figure 2, are as follows:

- Requirements Analysis
- Package Identification, Evaluation and Selection
- Non-COTS Development
- Glueware Requirements and Development
- System Integration and Test
- Target System Installation and Acceptance Test
- Discrepancy Resolution
- Sustaining Engineering

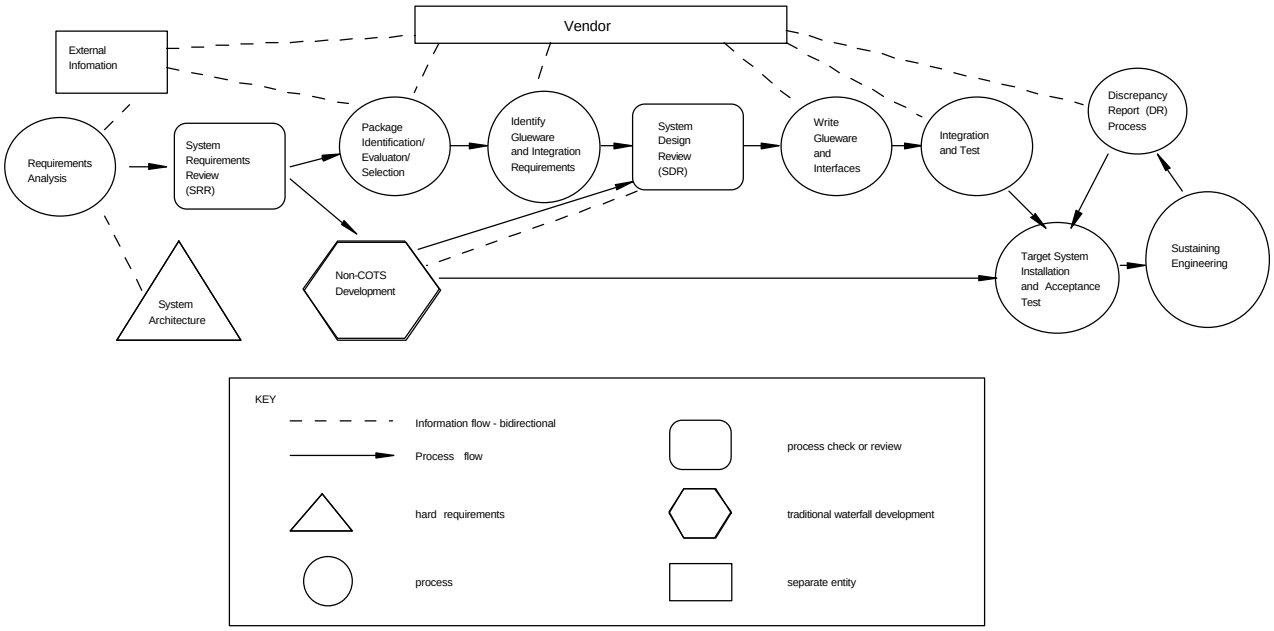

Figure 2. Process Flow for COTS Projects

The earliest steps in COTS-based development are similar to traditional development - requirements gathering. In the requirements phase a strong emphasis is on gathering external information. Much of this information comes from separate organizations, particularly the product vendor, in the form of documented functionalities. Some project requirements are predefined, with minimal requirements analysis needed. Early reviews of the requirements are crucial even with a less formal process.

Following requirements analysis are the new and concurrent steps of package identification, evaluation, and selection. These are new activities, requiring new technical skills and new administrative duties, especially in the area of procurement.

Package identification consists of Web searches, product literature surveys and reviews, other system component reuse, and recommendations from external sources. Product

information is kept in a central justification notebook, or an evaluation notebook. Not only are product evaluation notes kept, but subjective comments concerning the vendor quality and responsiveness are kept, too.

As packages are identified, the evaluation and selection processes begin. Package evaluation steps mentioned in the interviews consisted of prototyping, vendor demonstrations, and in-depth review of literature such as manuals and user guides. Glueware and interfaces as dictated by the system architecture, operating system and hardware are identified. Vendor training, sites, and availability, are considered. Procurement issues surface such as development fees for added requirements, licensing and maintenance fees and sustaining engineering support.

The selection step sometimes uses a weighted average. To do this, vendor capabilities are listed and mapped to the system requirements. With team agreement, weights of importance are assigned to each requirement. Then each team member votes. Team members are polled and the votes tallied. Discussion ensues and a choice is made. In cases where the vendor will code additional functionality, the vendor is notified of the decision. In the case of one team, when the vendor was told they were selected, the vendor announced a hidden cost. Negotiations ended altogether, and the second choice vendor and package were used.

In both of these first two process stages, we found that some projects relied on the COTS Evaluation Team, which is chartered by the parent organization to survey the marketplace and evaluate vendor packages that fall within the domain expertise of the mission team's organization. The evaluation team then reports its findings and offers this knowledge to the project teams. The project team is ultimately responsible for deciding what package to select and integrate. The evaluation team can be important when delivery time is driving the project - time the development team doesn’t have for product evaluations.

Most projects studied have an element of traditional development that does not depend on COTS or other packages. This development begins in parallel with the early COTS-related steps, as a traditional development project. Non-COTS cost and schedule are monitored. There is a bi-directional information flow between the COTS-based process flow and the non-COTS development that comes into play in the design review. Only some teams held a formal System Design Review (SDR), but all teams mentioned some mechanism to apprise the customer of the design.

After the design review, whether it is formal or informal, traditional non-COTS development continues in parallel with the coding of the glueware and the interfaces. Close contact with the vendor technical staff, or a competent Help Desk is essential during this development.

The integration step varies a great deal from project to project, depending on which and how many COTS products are being used. At system integration and testing the COTS packages are treated as black-boxes. The teams commented that testing focused on the

interface glueware and the input file format. Again, the importance of the vendor technical staff or Help Desk availability was emphasized. Testing is conducted on each software component as the components are integrated, piece-by-piece.

Unlike the traditional life-cycle, no formal acceptance testing or operational readiness reviews were mentioned by the teams. The development team installs the software on the target system. Once installed, navigational training to familiarize the customer with the system is conducted. During this phase, a member of the development team is the single point-of-contact or intermediary between the customer and the vendor. This person is responsible for reporting discrepancies, and handling software "patches" or corrections. Interviewees mentioned that software patches were placed on vendor Web sites that were downloaded to the target system.

The end of the configuration process is marked by the sustaining engineering effort. To date no team that the study team interviewed had reached the sustaining engineering stage.

4.2 Lessons Learned & Experience Gained

The developers interviewed were also asked to describe the major differences between COTS-based development and traditional development, and the advantages and drawbacks. Some mentioned the obvious difference, i.e. that there is now a whole lot of software that doesn't need to be implemented. It's no longer the task of building a big system, but of using already-built pieces. But there were other less obvious differences. Some of those differences mentioned were:

- different design phases
- looser process requirements
- new or greatly increased need for vendor interaction
- procurement skills now needed
- new or greatly increased need for product evaluations
- no unit test or inspections of packaged software

Advantages of COTS-based development that were mentioned included[*]:

- more flexible requirements
- less process overhead
- less code to write
- less debugging
- shorter cycle time
- better adherence to schedule.
- serendipitously useful functionality in COTS packages

[*] Note: "Shorter cycle time" and "less process overhead" may be due to the pressure to do things faster as much as due to the adoption of COTS.

Many of the disadvantages mentioned had to do with dealing with the vendor, including the risks of less than full knowledge beforehand, dependence on the vendor, and vendor negotiations. Another disadvantage, which some people listed as an advantage, is the relative looseness of the process in package-based projects. Some people thought that more rigor was needed.

5 Packaging the Approach

FDD projects have moved rapidly from a reuse-based development process to a COTS-based system development process. The SEL needed to react quickly with new mechanisms to adapt to these and other changes in the environment. The changes that the SEL has undergone are important to study because the nature of software development is changing and will require further changes in the research methods of the SEL and other organizations. As we encounter new problems, we need new ways to address these issues.

We will address the natural adaptation of the SEL that is taking place; the learning, refitting and adjusting of the SEL learning procedures in order to keep pace with the new organization and environment. Three major innovations in standard procedures will be discussed :

1) The use of qualitative analysis, mostly in the form of structured interviews and analysis of data gathered from those interviews.
2) Changes to the database in terms of what is being collected and analyzed in order to keep track of the changing business in FDD.
3) The generation of *Study Briefs*, which are short, quickly disseminated communications on a variety of topics—lessons learned, early analysis results, definitions of new terms, etc.—to keep information flowing between the EF and the project organization in a timely manner.

The use of qualitative analysis was necessitated by the COTS study. The study team found that with this change in technology, the quantitative data that the SEL collects does not tell the entire story of what is occurring on projects. During the course of these interviews the SEL team members interacted with the technical personnel. These interactions led the SEL to realize the need for more effective, frequent communication: (1) communication from the SEL to the project organization about what the SEL was learning, and (2) feedback from the project organization to the SEL to corroborate and refine the SEL's evolving models. This realization became the catalyst for the SEL *Study Briefs*. Interviews specific to the COTS study also showed that the data collected for COTS was insufficient. This sparked the modification to the Weekly Effort Form to include COTS specific details. The transition to this new form has been simple due to the new re-engineered SEL database, that has been revolutionized using COTS products and transitioned to a workstation platform. This allows us to use the database as a repository for information on the COTS products used as well as the effort involved in putting together a COTS based system.

5.1 Interviews and Qualitative Data

Empirical studies in software engineering, like the ones that the SEL has engaged in for two decades, have traditionally relied on standard quantitative methods in order to characterize some aspect of a software development process. In some cases, several quantitative studies of various sizes and scopes have been conducted to address one general issue, e.g. Cleanroom software development [Selby et al., 1987]. Approaching a problem from several angles in this way yields a more complete description of a particular process or of the effect of a particular technology. This approach has helped the SEL and other organizations learn a great deal about their software business. In recent years, however, software projects in the SEL environment have become both more complex and faster-paced, as is true in much of the software industry. This has motivated the SEL to find ways to provide richer answers to more complex problems in less time.

One approach to achieving this is to use different research methods than the SEL is accustomed to using, in particular qualitative methods. Qualitative data is information in the form of words and pictures, as opposed to quantitative data, which is in the form of numbers. Qualitative analysis is simply the examination and analysis of qualitative data in order to form conclusions and hypotheses. Qualitative data is by definition richer and carries more information than quantitative data. On the other hand, it is more complex and harder to analyze. Qualitative analysis methods have been designed to deal with this complexity [Glaser and Strauss, 1967]. Combinations of qualitative and quantitative methods are especially useful because the two types of methods tend to deal with the complexity of the subject in complementary ways.

The COTS study is one of the first SEL studies to use qualitative data to a large extent. The qualitative data used in this study comes from extensive interviews with software developers and managers. Using this data has allowed an in-depth examination of COTS-based development that incorporates a variety of perspectives in one study. For example, data was collected on the problems encountered during COTS-based development, the different steps involved, the parts of the process which are effort-intensive, and the roles that must be filled to carry out this type of development. Much of this information would be very difficult to collect quantitatively, and would have required multiple studies, each measuring various attributes in different ways.

The drawbacks to doing qualitative study is that it doesn't provide "hard" results in terms of easy-to-use mathematical models (e.g. regression models) or easy-to-summarize relationships between variables (e.g. correlations). Instead, qualitative results are more complex, "messier", to reflect the complexity of the problem being described.

Qualitative data, mostly from interviews, is also being used to some extent on other ongoing SEL studies. In combination with other quantitative methods, we believe the use of qualitative analysis in current and future studies will help the SEL provide the development community with more useful, in-depth, and realistic explanations of software development phenomena.

5.2 New Data Forms - Quantitative Data

In response to a need for more COTS related data, the SEL realized an opportunity to update the types of data that are maintained in the SEL database. This was accomplished by the modification of an existing form, the Weekly Effort Form, and the addition of a new form, the COTS & Tools Information Form.

5.2.1 Weekly Effort Form

As the interview data was leading us to define the COTS-based development process, the study team saw that there were new activities that projects were conducting. These include:

COTS/GOTS Evaluation
COTS/GOTS evaluation activities included identifying packages, collecting information, attending demos, evaluating and selecting COTS/GOTS packages.

COTS/GOTS Integration
This included integrating COTS/GOTS, possibly with other software components, to produce individual applications or subsystems. This also included the writing and debugging of glueware.

COTS Package Familiarization
Package familiarization is spending time to learn to use a COTS package, not including formal training, which would be included under other effort categories, nor package familiarization for the purposes of evaluation.

Configuration Management
Configuration management had not previously been a separate category.

Procurement
This included procuring and purchasing packages, interacting with the vendor regarding licensing and maintenance agreements, etc.

These new activities were merged into the Weekly Effort Form (WEF), the existing SEL form for collecting effort data from the technical personnel. This merger created a WEF modified for COTS that was then used on a trial basis by two projects. (See Appendix A for the original WEF and Appendix B for the experimental COTS WEF.) After experimental use of this COTS WEF, and a few resulting updates, the SEL decided to implement the updated WEF across the organization. This was accomplished through full consultation with FDD technical personnel. The resulting WEF was put into place November 1997 across the organization (see Appendix C).

The graph shown in Figure 3 indicates the type of data collected by the experimental COTS WEF, the WEF which was introduced in October 1995 (and which had only a single COTS activity category), and the even earlier SEL weekly effort form which was in use prior to October 1995. The leftmost bar shows the typical distribution of effort on

completed FDD projects prior to October 1995. The major activities are *design*, *code*, *test*, and *administrative*; none deal with COTS.

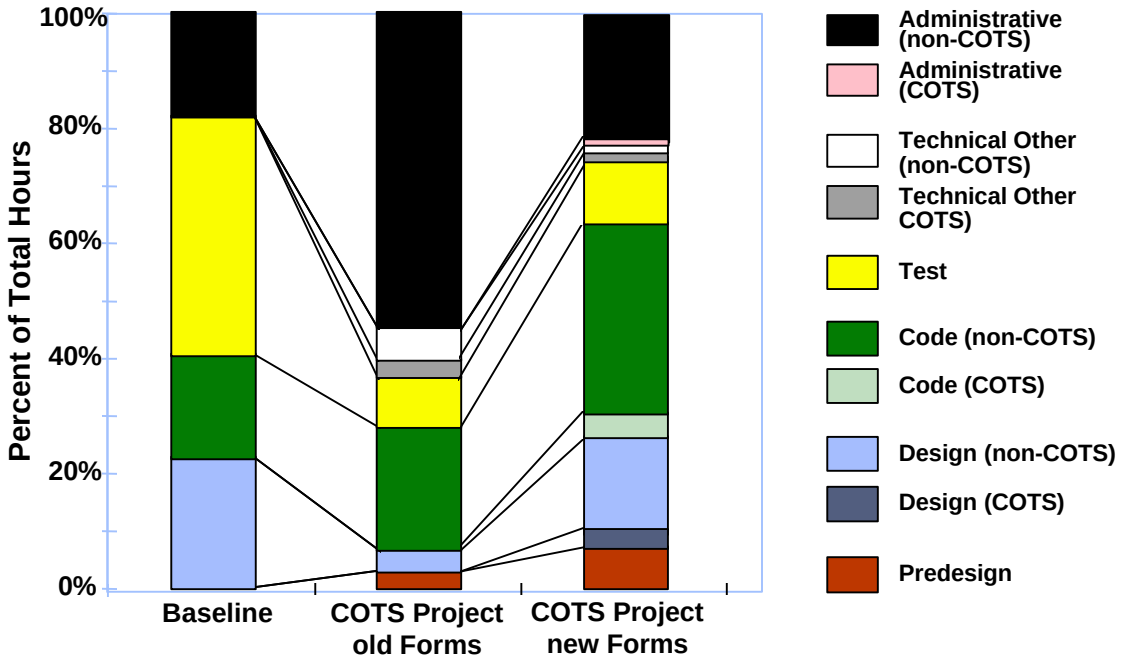

Figure 3. COTS Data From Projects

The middle bar shows the effort distribution for a nearly complete FDD project that was developing during the era of the WEF that was introduced in October 1995 and which involved some COTS integration. This WEF introduced a *predesign* category. It also introduced a single *COTS* activity and a so-called *technical other* category. Note that this bar shows a great increase in the proportion of project effort spent in the *administrative* activity. Various hypotheses were examined to explain this change, but none proved conclusive.

The rightmost bar shows the distribution of effort for a FDD project that involved a fair amount of COTS integration but which was only partially complete. This project began using the experimental COTS WEF soon after the project began. Only about twelve weeks of this project's effort data were available for analysis for this paper. The data in this bar is thus insufficient to draw any conclusions on the distribution of effort on a typical FDD project, yet alone a project in another environment. Data on several complete projects would be required before the typical FDD effort distribution on a COTS project could be determined.

Future studies are underway to determine this and to address more specific issues. These are described in Section 6.

5.2.2 COTS & Tools Information Form

In order to collect context data about the COTS packages used on projects, the SEL developed the COTS & Tools Information Form (CTIF), shown in Appendix D. The need for the CTIF became evident during the interview process. We were collecting qualitative data, such as which COTS packages are used, what support is provided by the vendor, and whether it is embedded into the system or merely a tool. Rather than maintaining all this information in the interview notes, we developed the CTIF to collect data that would be stored, and readily accessible, in the SEL database. Using the CTIF to collect this context data allows us to characterize the COTS products in order to better compare projects that are related either in the type of COTS products used, or in functionality provided by COTS.

5.3 Study Briefs

The SEL realized a need for compact products. *SEL Study Briefs* are an example of this as they concisely document and distribute information that might fall through the cracks. A Study Brief is less than a process document, yet much more than informal communications. The modularity of the Study Briefs allows the user community to incorporate "one page worth of process" into their busy schedules. Study Briefs also serve as a tool for communication with the inclusion of the technical community in the feedback loop section. A sample Study Brief is shown in Appendix E.

The format of a SEL Study Brief is shown Table 1.

Study Brief Field	Explanation of Field
Study Brief Number	number assigned (in order of posting on web)
Issue	topic of Study Brief
Purpose	goal of Study Brief
Current Understanding	body of the Study Brief, may vary in style
Feedback	comments received from audience after Study Brief has been posted to web site
Original Author	person or persons originating Study Brief
Responsible Author	person responsible for receiving feedback, and possibly modifying the Study Brief (in most cases same as Original Author)
Contributors	others who contributed to the Study Brief
References/Relevant Links	materials used in preparation of Study Brief, additional information on a topic, or hot links to on-line sources
History	first published date, any revision dates

Table 1. SEL Study Brief Format

6 Future Directions

After analysis of the current process and review of the issues that are most relevant to this new COTS-based development environment, several topics for further study have been identified:

1) The long-range effects of COTS use, in particular the maintenance of systems which incorporate COTS packages,
2) Modeling and estimating effort, cost, and schedule of COTS projects based on data collected with the new forms,
3) Risks of COTS use, to be studied with a series of SEL case studies, possibly including the re-engineered SEL database development, and
4) Methods for measuring the "distance" between a set of requirements for a new system and the available COTS packages which could be used to satisfy those requirements.

The selection and implementation of COTS is easier to understand than is the maintaintenance of a COTS-based system. Such a system will require modifications and enhancements during its lifetime, and many of these modifications may be prompted by vendor updates to the COTS packages. Maintenance includes less obvious costs; maintenance agreements and licensing are more tangible than the effort that will be required to identify the parts of the code that are affected by a small change elsewhere so that the modification to one area does not cripple the system. The projects interviewed for the COTS study had not moved into a maintenance phase. The SEL sees this as an area for further research.

The SEL has a long-standing tradition in the FDD for providing models for the estimation of effort, cost and schedule. The interviews uncovered a need for new models to support COTS projects. The SEL has begun efforts to baseline the current situation across the organization. The next steps toward developing models include collecting a reasonable amount of data from which to draw quantifiable conclusions.

The interviews identified risk as an important topic. The SEL determined that gathering case studies of various COTS-based systems with emphasis on the risks expected, as well as the risks involved, would provide valuable information. It appears likely that many of the risks of introducing COTS systems are domain-independent. Because of this the SEL's recently re-engineered software metrics collection and reporting system would be a good non-FDD system to examine as a case study of COTS risks. In this re-engineering process, the SEL upgraded the SEL's COTS relational database management system and added an additional COTS product to automate the submission of data to the SEL by users.

In related research, we seek to develop a mechanism to measure the "distance" between the need for functionalities (the requirements) and the specification of available COTS. Such "functional distance" measures should help us to predict the amount of glueware necessary to integrate COTS with the rest of the system. The costs of glueware is one key factor in the total cost of using COTS software.

Appendix A - Old WEF (Introduced October 1995)

WEEKLY EFFORT FORM

Use this form to record all hours you worked during the week.

Name:

Project:

Date (Friday):

For Librarian's Use Only	
Number:	
Date:	
Entered by:	
Checked by:	

Application or Subsystem (or "N/A" as appropriate.)						
Release/Build Number (or "N/A" as appropriate.)						
SCR Number (or "N/A" as appropriate)						
Hours By Activity (List hours in a separate column for each application, release/build and SCR combination.)						
PRE DESIGN	Requirements Spec. Definition/ Development	Hours spent defining and developing the requirements specifications				
	Requirements Analysis	Hours spent understanding requirements specs or understanding SCRs for enhancements or adaptations				
	Error Analysis/ Debugging	Hours spent finding a known error in the system; may be in response to SFR, STR, SCR. (includes generation and execution of tests associated with finding the error)				
	Impact Analysis/Cost Benefit Analysis	Hours spent analyzing several alternative implementations and/or comparing their impact on schedule, cost, and ease of operation				
DESIGN	Design Creation or Modification	Hours spent developing or changing the system, subsystem, or component design (includes development of PDL, design diagrams, meeting materials, etc.)				
	Design Review/ Inspection	Hours spent reading or reviewing design (includes design meetings and consultations, as well as formal and informal reviews, walkthroughs, and inspections)				
CODE	Code Generation/ Modification	Hours spent actually coding system components (includes both desk and terminal code development)				
	Code Review/ Inspection	Hours spent reading code (for any purpose other than isolation of errors) or inspecting other people's code				
	Unit Testing	Hours spent testing individual components of the system (includes writing test drivers and informal test plans)				
TEST	System Integration/ Integration Testing	Hours spent integrating components into the system; hours spent writing and executing tests that integrate system components (includes system tests)				
	Regression Testing	Hours spent regression testing the modified system				
	Independent Testing Support	Hours spent supporting independent testing, including training of testers				
MISC	Prototyping	Hours spent prototyping to investigate a particular issue (not to be confused with other activity hours when the entire system is a prototype)				
	COTS/GOTS	Hours spent evaluating, selecting, procuring, integrating and testing COTS/GOTS products				
OTHER	Documentation	Hours spent creating and reviewing deliverable documents				
	Training for Self	Hours spent taking courses (including computer-based training), attending seminars, etc.				
	User Support/Training	Hours spent training users and responding to their questions				
	Management	Hours spent managing or coordinating work and reporting status				
	Other	Other development hours not covered above				
	Total	Total hours per column	0.0	0.0	0.0	0.0
	Grand Total	Total hours	0.0			

Appendix B - Experimental COTS WEF

Experimental "COTS modified" **WEEKLY EFFORT FORM**

Use this form to record all hours you worked during the week.

Name:

Project:

Date (Friday):

For Librarian's Use Only	
Number:	
Date:	
Entered by:	
Checked by:	

Application or Subsystem (or "N/A" as appropriate.)						
Release Number (or "N/A" as appropriate.)						
Build Number (or "N/A" as appropriate.)						
SCR Number (or "N/A" as appropriate)						
Hours By Activity (List hours in a separate column for each application, release/build and SCR combination.)						
PREDESIGN	Requirement Specs. Definition/Development	Hours spent defining and developing the requirements specifications				
	Requirements Analysis	Hours spent understanding requirements specs or understanding SCRs for enhancements or adaptations				
	Error Analysis/ Debugging	Hours spent finding a known error in the system; may be in response to SFR, STR, SCR.				
	Impact Analysis/Cost Benefit Analysis	Hours spent analyzing several alternative implementations and/or comparing their impact on schedule, cost, and ease of operation				
DESIGN	COTS/GOTS Evaluation	Hours spent in COTS/GOTS evaluation activities, (i.e., identifying packages, collecting information, attending demos, evaluating & selecting COTS/GOTS packages)				
	Design Creation or Modification	Hours spent developing or changing the system, subsystem, or component design (includes development of PDL, design diagrams, meeting materials, etc.)				
	Design Review/ Inspection	Hours spent reading or reviewing design (includes design meetings and consultations, formal and informal reviews, walkthroughs, and inspections)				
CODE	COTS/GOTS Integration	Hours spent integrating COTS/GOTS (& other software components) to produce individual applications/ subsystems (i.e. writing & debugging glueware, COTS package familiarization)				
	Code Generation/ Modification	Hours spent actually coding system components (includes both desk and terminal code development)				
	Code Review/ Inspection	Hours spent reading code (for any purpose other than isolation of errors) or inspecting other people's code				
	Unit Testing	Hours spent testing individual components of the system (includes writing test drivers and informal test plans)				
TEST	System Integration/ Integration Testing	Hours spent integrating components into the system; hours spent writing and executing tests that integrate system components (includes systest)				
	Regression Testing	Hours spent regression testing the modified system				
	Indep. Testing Support	Hours spent supporting independent testing, including training of testers				
MISC	Procurement	Hours spent procuring/purchasing, interacting with vendor regarding licensing/maintenance agreements etc.				
	Prototyping	Hours spent Prototyping to investigate a particular issue				
OTHER	Documentation	Hours spent creating & reviewing deliverable documents				
	Training for Self	Hours spent taking courses (including computer-based training), attending seminars, etc.				
	User Support/Training	Hours spent training users and responding to their questions				
	Configuration Mgmt.	Hours spent in configuration management				
	Management	Hours spent managing or coordinating work and reporting status				
	COTS/GOTS Other	Other COTS/GOTS specific hours not covered above				
	Other	Other development hours not covered above				
	Total	Total hours per column	0.0	0.0	0.0	0.0
	Grand Total	Total hours	0.0			

Appendix C - New WEF (Introduced November 1997)

WEEKLY EFFORT FORM

For Librarian's Use Only

Number: _____

Date: _____

Entered by: _____

Checked by: _____

Use this form to record all hours you worked during the week.

Name:

Project:

Date (Friday):

Application or Subsystem (or "N/A" as appropriate.)						
Release Number (or "N/A" as appropriate.)						
Build Number (or "N/A" as appropriate.)						
SCR Number (or "N/A" as appropriate.)						
Hours By Activity (List hours in a separate column for each application, release/build and SCR combination.)						
PREDESIGN	Requirement Specs. Definition/ Development	Hours spent defining and developing the requirements specifications				
	Requirements Analysis	Hours spent understanding requirements specs or understanding SCRs for enhancements or adaptations				
	Error Analysis/ Debugging	Hours spent finding a known error in the system; may be in response to SFR, STR, SCR (includes generation and execution of tests associated with finding the error)				
	Impact Analysis/ Cost Benefit Analysis	Hours spent analyzing several alternative implementations and/or comparing their impact on schedule, cost, and ease of operation				
DESIGN	COTS/GOTS Evaluation	Hours spent in COTS/GOTS evaluation activities (i.e., identifying packages, collecting information, attending demos, evaluating and selecting COTS/GOTS packages)				
	Design Creation or Modification	Hours spent developing or changing the system, subsystem, or component design (includes PDL, design diagrams, meeting materials)				
	Design Review/ Inspection	Hours spent reading or reviewing design (includes design meetings and consultations, formal and informal reviews, walkthroughs, and inspec.)				
CODE	COTS/GOTS Integration	Hours spent integrating COTS/GOTS (may be with other software components) to produce individual applications/ subsystems (i.e., writing and debugging glueware)				
	Code Generation/ Modif.	Hours spent actually coding system components (desk & terminal dev.)				
	Code Review/ Inspection	Hours spent reading code (for any purpose other than isolation of errors) or inspecting other people's code				
	Unit Testing	Hours spent testing individual components of the system (includes writing test drivers and informal test plans)				
TEST	System Integration/ Integration Testing	Hours spent integrating components into the system or writing and executing tests that integrate system components (includes sys-test)				
	Regression Testing	Hours spent regression testing the modified system				
	Indep. Testing Support	Hours spent supporting independent testing, including training of testers				
OTHER	COTS Package Familiarization	Hours spent learning to use a COTS package (not formal training, which would be listed under training for self; also does not include evaluation)				
	Prototyping	Hours spent prototyping to investigate a particular issue				
	Training for Self	Hours spent taking courses (including computer-based training), attending seminars, etc.				
	User Support/Training	Hours spent training users and responding to their questions				
	CM	Hours spent in configuration management				
	Procurement	Hours spent procuring/purchasing, interacting with vendor regarding licensing/maintenance agreements, etc.				
	Documentation	Hours spent creating and reviewing deliverable documents				
	Management	Hours spent managing or coordinating work and reporting status				
	COTS/GOTS Other	Other COTS/GOTS specific hours not covered above				
	Other	Other hours not covered above (i.e., department and all-hands mtgs)				
	Total	Total hours per column	0.0	0.0	0.0	0.0
	Grand Total	Total hours	0.0			

Appendix D - CTIF

COTS & TOOLS INFORMATION FORM (CTIF)

Use this form to obtain context and evaluation data, verify at project completion.
For each COTS product or Tool, use a separate CTIF.

For Librarian's Use Only
Number:
Date:
Entered by:
Checked by:

Name: **Date:**

Project:

COTS Product or Tool: **Version Number:** **Vendor:**

1. Reasons for using tool or COTS: Check all that apply.
   ☐ requirements definition ☐ requirements analysis ☐ requirements ☐ tracking/traceability ☐ design
   ☐ simulation/modeling ☐ code generation ☐ static analysis ☐ compilation ☐ debugging
   ☐ configuration management ☐ integration ☐ QA ☐ re-engineering ☐ testing
   ☐ reverse engineering ☐ change management ☐ project tracking ☐ documentation
   ☐ training ☐ information management ☐ reuse management ☐ measurement ☐ risk analysis ☐ communication ☐ project planning/estimation ☐ application functionality

2. Support provided for tool or COTS: Check all that apply.
   ☐ demos ☐ informal or partial documentation ☐ full documentation ☐ courses ☐ help desk

3. Activities supported by tool or COTS: Check all that apply.
   ☐ requirements definition ☐ requirements analysis ☐ design ☐ coding ☐ testing
   ☐ documentation ☐ CM ☐ QA ☐ management ☐ other

4. Usage frequency of tool or COTS: (Select one from choices below, enter letter of selected item here.)
   a. no usage b. used once or twice c. monthly d. weekly e. daily

5. Functionality of tool or COTS: (Select one from choices below, enter letter of selected item here.)
   a. no data available b. abandoned, due to lack of functionality c. major expected functions missing
   d. some expected functions missing e. most expected functions present f. all expected functions present

6. Usefulness of tool or COTS: (Select one from choices below, enter letter of selected item here.)
   a. no data available b. abandoned, due to problems c. many problems encountered
   d. some problems encountered e. few problems encountered f. no problems encountered

7. Impact of tool or COTS on project's success: (Select one from choices below, enter letter of selected item here.)
   a. impossible to estimate b. major negative impact c. some negative impact overall
   d. positive & negative impacts balance out e. some positive impact f. major positive impact

8. Is COTS or tool embedded in software, i.e., is COTS being delivered as part of the system? YES ☐ NO ☐

Appendix E - Sample SEL Study Brief

Study Brief Number: 7

ISSUE: COTS Evaluation Team

PURPOSE: Document the SEL's understanding of the COTS Evaluation Team, for the purpose of disseminating information to the FDF community and clarification for the SEL, in regards to the COTS Study.

CURRENT UNDERSTANDING:

Team was formed in 1995 to address a move towards COTS solutions in FDD. Originally, part of Code 551, Flight Mechanics. Currently, part of the Code 550 Flight Dynamics Technical Support Office (TSO).

Who is the Evaluation Team?

- Composed of problem domain experts and mission team members, Led by Sue Hoge, GSFC analyst
- Matrixed on a as needed basis, not dedicated full-time to evaluations

What are they doing?

- Evaluate COTS for Flight Dynamics Mission Planning & Orbit Determination
- Provide evaluation services to mission teams, as requested
- Provide independent software evaluations
- Monitor new COTS products, as available/maintain data on products that meet specific domain needs
- Publish Evaluation Reports
- Update the Guidelines for Evaluating COTS at the FDF document, as needed

What process is followed?

Basic process is outlined in the Guidelines for Evaluating COTS at the FDF document

- Establish the Objectives of an Evaluation
- Establish the Evaluation Type
- Determine the Evaluation Method
  - Basic/Standard Evaluation Methods
  - Variations on the Standard Evaluation Methods
- Establish Evaluation Criteria
- Perform Evaluation
- Document Results
- Benchmarks/Regression Testing/Follow up Evaluations

What have they evaluated?

TK (AGI)
PODS (AGI)
GEODYN(Code 900, GOTS)
OASYS(ISI)
PROBE(BBN)
PATTERN (BBN)
GREAS (AGI)

What problems have been encountered?
Public awareness of Evaluation Team and services is low

What else we learned about the Evaluation Team?
They are building an "experience base" of COTS evaluations (for multiple products, multiple missions).

Guidelines document:

Domain specific, and not intended to be a general methodology for any COTS s/w evaluation
Working document with lessons learned mixed in with process
Written from a hands-on perspective

What do we suggest?
The COTS Product Evaluation Questions from SEL Packaged-Based System Development document (page 22, table 3) are valid in the COTS Evaluation Team environment. Recommend that the SEL distribute the modified COTS Product Evaluation Questions (addition of two questions suggested by Sue Hoge) as a "One Pager" to technical personnel. Recommend that the Evaluation team use modified COTS Product Evaluation Questions as part of their process, since these are issues that Sue Hoge typically addresses with Evaluation Team.

FEEDBACK: (none available at this time, email comments to responsible author)

ORIGINAL AUTHORS: Amy Parra and Steve Kraft
RESPONSIBLE AUTHOR: Amy Parra
CONTRIBUTORS: Sue Hoge

REFERENCES/RELEVANT LINKS:

- Guidelines for Evaluating COTS at the FDF document
- STK Evaluation and Test Results
- STK PODS Evaluation Final Report
- OASYS Evaluation Report
- Interview notes (from two COTS Study interviews with Sue Hoge)

HISTORY: Study Brief published 11/11/97.

Appendix F - Interview Guides

Interview Guide 1a: Initial Project Interviews

Who: project leads
Subjects covered: background and current status of project, GSS vs. MATLAB decisions, initial COTS information
Duration: 30-45 minutes

Note: This interview should also include introducing ourselves and our study to the project leads.
Interviewee:

Interviewer:

Scribe:

Date of interview:

Duration:

Location:

1. What is/are your ROLE(s) on this project (get both official titles, e.g. user, domain expert, as well as a decription, e.g. technical vs. administrative, level of involvement, etc.)?

2. What is the current status of FDSS development for this project? What are the different applications being developed? Which have begun, are in progress, or are completed? [gradually narrow down to attitude applications]

3. For each application, how is it being developed? Using GSS and UIX? Using some COTS product like MATLAB or STK? Did any modifications need to be made to the COTS or GOTS products? Describe the modifications and how they were made.

4. What deployment/development/integration process did you use to produce these applications? Where did this process come from? What process documentation or guidance did you use, if any?

5. Are you aware of the SEL PACKAGED-BASED SYSTEM DEVELOPMENT PROCESS DOCUMENT?

6. Did you follow the SEL PACKAGED-BASED SYSTEM DEVELOPMENT PROCESS DOCUMENT?

7. Is there anything that we can do to make this a more useful, easier-to-follow process?

8. How were the decisions to use these COTS and GOTS

products made? What were the steps in the decision process? What were the criteria?

9. Were lessons learned recorded? Where?

10. What types of problems did you run into with the COTS and GOTS products you chose?

11. What do you think are the biggest risks associated with these decisions? [try to get a mapping between the criteria mentioned in #3, and the risks mentioned here] For example:

- unacceptable performance of the application,
- reliability of COTS products,
- delays waiting for something from another group,
- delivered application is unmaintainable,
- required skills not available
- key personnel leaving or being pulled off project at crucial points
- cultural clashes between personnel from different areas
- turnaround time for error fixes or added functionality

12. Any creative ways to protect against these risks?

13. What data did you collect during the project regarding COTS?

〈 schedule
〈 cost
〈 errors
〈 standard SEL data

14. What metrics do you see as valuable in managing COTS-based projects?

15. Was there a purchasing leader for this project, who? (discuss purchasing decisions, procurement)

16. What other projects do you know are using or planning to use COTS, GOTS, or other package-based products?

17. Can I be put on your project mailing list and/or could I have access to your project Web page? What else would help me keep track of how the project is going? Where can I look at project documentation?

18. Who are the other core team members and what are their roles?

Interview Guide 2: COTS Follow-Up Interviews

Who: COTS-based project leads
Subjects covered: Follow Up COTS information
Duration: 30-45 minutes

Note: This interview should also include re-introducing ourselves and our study to the project leads.

Interviewee:

Project(s):

Interviewer:

Scribe:

Date of interview:

Duration:

Location:

Date of initial interview:

1. What did you do for the following: (try to capture the major activities, process, products, reviews)

   a. Requirements Analysis
   b. Package Identification
   c. Architecture Definition
   d. Package Selection
   e. System Integration
   f. Test
   g. Maintenance

2. What are the biggest differences between traditional development and Package-Based Development?

3. What are the advantages of Package-Based Development in comparison with traditional development?

4. What are the disadvantages of Package-Based Development in comparison with traditional development?

5 Are you familiar with the SEL Package-Based System Development Process document, Feb. 1996?

6. For an upcoming COTS-based project would you use the SEL Package-Based System Development Process?

   a. If yes, why
   b. If no, why not

7 What parts of the process and/or the document would you improve and how?

Interview Guide 3: Additional COTS Follow-Up Interviews

Who: COTS-based project leads
Subjects covered: Follow Up COTS information
Duration: 15 minutes

Note: This interview should also include re-introducing ourselves and our study to the project leads. Mention that this final interview is to verify the data we have collected, and clarify any areas on which we needed more information. For this interview, meet with the project lead and any other team members that you think would be appropriate to include, to verify all the data collected on that project.

Interviewee(s):

Project(s):

Interviewer:

Scribe:

Date of interview:

Duration:

Location:

Date of initial interview:

Date of follow- up interview:

Before the interview:
List the CTIFs that are on the Kano Drive for that project
Verify the matrix and supply any reasons why process steps were or were not followed

Bring to the interview:
Matrix for that project
Process Characterization

Actual Interview Questions:

1. Have you completed CTIFs for each COTS or Tool that you are currently using? (definitely for all SEL projects, ask non-SEL project to also comply)

If not, fill in hard copies of CTIFs during the interview with the project lead.

2. This is the process characterization that we have developed after interviewing projects. How representative is it of your project? (take notes as to areas that they believe they differ from the process characterization)

3 These are the specific process steps that we noted during interviews. (Show matrix of Steps vs. Interviews for that project) Allow me to review the data that we have from you as to whether or not you followed a certain process step. Fill in YES for project completed this step, fill in NO for project did not do this step. Give a simple reason for why the project completed or did not complete a step.

References

Condon, S., C. Seaman, V. Basili, S. Kraft, J. Kontio, and Y. Kim. "Evolving the Reuse Process at the Flight Dynamics Division (FDD) Goddard Space Flight Center." *Proceedings of the Software Engineering Workshop*, NASA/Goddard Space Flight Center, pp. 27-42, December 1996.

Glaser, B.G. and A.L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Company, 1967.

SEL Recommended Approach to Software Development, Revision 3. Software Engineering Laboratory Series, SEL-81-305, June 1992

Selby, Richard, Victor R. Basili, and Terry Baker. "Cleanroom Software Development: An Empirical Evaluation." *IEEE Transactions on Software Engineering*, pp. 1027-1037, September 1987.

Waligora, S. *Packaged-Based System Development Process*. Software Engineering Laboratory, February 1996.